Nonparametric Decoding for Generative Retrieval

Hyunji Lee[1] **Jaeyoung Kim**[3*] **Hoyeon Chang**[1] **Hanseok Oh**[1] **Sohee Yang**[1]
Vlad Karpukhin[2] **Yi Lu**[2] **Minjoon Seo**[1]

[1] KAIST AI [2] Forethought.AI [3] Kakao

{hyunji.amy.lee, hanseok, sohee.yang, retapurayo, minjoon}@kaist.ac.kr
{vlad.karpukhin, yi.lu}@forethought.ai jay.eong@kakaocorp.com

Abstract

The generative retrieval model depends solely on the information encoded in its model parameters without external memory, its information capacity is limited and fixed. To overcome the limitation, we propose Nonparametric Decoding (Np Decoding) which can be applied to existing generative retrieval models. Np Decoding uses nonparametric contextualized vocab embeddings (external memory) rather than vanilla vocab embeddings as decoder vocab embeddings. By leveraging the contextualized vocab embeddings, the generative retrieval model is able to utilize both the parametric and nonparametric space. Evaluation over 9 datasets (8 single-hop and 1 multi-hop) in the document retrieval task shows that applying Np Decoding to generative retrieval models significantly improves the performance. We also show that Np Decoding is data- and parameter-efficient, and shows high performance in the zero-shot setting.[1]

1 Introduction

Text retrieval is often formulated as finding the most relevant items from a large corpus given an input query. The bi-encoder approach of using an encoder to map the documents and the query to a common vector space and performing a nearest neighbor search has been a common practice in text retrieval tasks (Karpukhin et al., 2020; Wu et al., 2020; Ni et al., 2021). Despite its high performance and popularity, it has an embedding space bottleneck (Luan et al., 2021; Lee et al., 2022); limited expressiveness due to fixed-size embeddings and misses the fine-grained interaction between embeddings as they interact in L2 or inner product space. Moreover, the bi-encoder approach requires large storage space to save all document embeddings.

A recently-proposed alternative to the bi-encoder approach is using a generative retrieval model (Cao et al., 2021; Tay et al., 2022; Bevilacqua et al., 2022; Lee et al., 2022; Wang et al., 2022; Lafferty and Zhai, 2003; Croft and Lafferty, 2010). It is an autoregressive model that retrieves the most relevant sequence by generating the target sequence (e.g., title, passage, document ID) token-by-token. It overcomes the embedding space bottleneck by interacting in the parametric space. Also, it is storage efficient by not having any external memory. However, the information capacity of such fully parametric models tends to be bounded by their sizes as it has to encode all information in its parameters (Tay et al., 2022; Roberts et al., 2020).

To this end, we propose Nonparametric Decoding (Np Decoding), a decoding method for generative retrieval models. It uses nonparametric contextualized vocab embeddings rather than vanilla vocab embeddings as decoder vocab embeddings. The contextualized vocab embeddings are output embeddings of an encoder that constructs a nonparametric dense vector space and are frozen during the training step whereas the vanilla vocab embeddings are trainable model vocab embeddings that construct a parametric space of the model. Therefore, by using Np Decoding, the generative retrieval model does not have to rely solely on its own parameters but can utilize the surrounding information encoded in the contextualized vocab embeddings (external memory). Note that while it utilizes the dense vector space as in the bi-encoder approach, unlike the approach, it does not have embedding space bottleneck as it is a variant of the generative retrieval model, and saves storage space by storing only clustering centroid embeddings (Section 3.5).

As shown in Figure 1, any generative retrieval model can incorporate Np Decoding by replacing the decoder vocab embeddings from the vanilla embedding matrix to contextualized embedding

*Work done during internship at KAIST AI.

[1]The code and datasets used in our work is at https://github.com/amy-hyunji/Contextualized-Generative-Retrieval.

Findings of the Association for Computational Linguistics: ACL 2023, pages 12642–12661
July 9-14, 2023

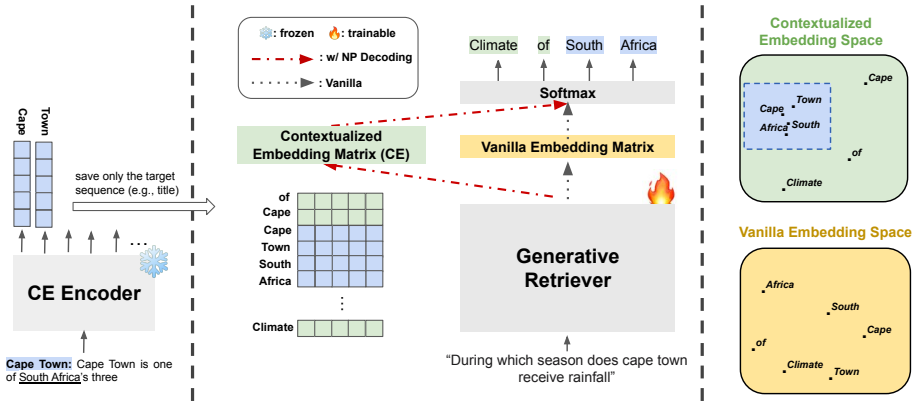

Figure 1: Np Decoding can be applied to any generative retrieval model by replacing the decoder vocab embeddings from the vanilla embedding matrix with the contextualized embedding matrix (CE). CE is composed of the output embeddings of the language model encoder (CE Encoder). Only the retrieval target sequences are added to CE, which in this figure we use the title (Cape Town) as the target sequence. Unlike vanilla vocab embeddings, contextualized vocab embeddings that consist CE contain context information, and a single token can have multiple token embeddings. This creates a more expressive and fine-grained contextualized embedding space compared to vanilla embedding space as shown on the right side of the figure.

matrix (CE) for both the training and the inference steps. By the replacement, Np Decoding has two key benefits over vanilla decoding. First, the generative retrieval model can utilize not only its parametric space but also its nonparametric space. The nonparametric space is constructed with decoder vocab embeddings of Np Decoding (CE), nonparametric and context-aware embeddings that capture surrounding information. Second, CE allows a token to have multiple token embeddings, unlike vanilla vocab embeddings where a token has a unique embedding. Therefore, the decoder vocab embedding space of CE becomes more expressive and fine-grained (right side of Figure 1). Since having a well-constructed CE is important for achieving high performance, we propose three different encoders (CE Encoder) used to output contextualized vocab embeddings added to CE (Section 3). We demonstrate that CE Encoder with contrastive learning results in a significant increase in performance.

The main contributions of our paper are as follows:

- We propose Nonparametric Decoding (Np Decoding), a simple and novel decoding method that can be applied to all existing generative retrieval models. Experimental results over 9 datasets show that Np Decoding can significantly improve the performance of existing generative retrieval models by leveraging both the parametric and the nonparametric space; 4.4% R-precision improvement for single-hop, 5.4% Recall@2 improvement for multi-hop datasets.
- We present various CE Encoder and show that training CE Encoder with contrastive learning further increases the performance by a large margin.
- We show generative retrieval models with Np Decoding are data- and parameter-efficient, and show higher performance in a zero-shot setting.

2 Related Work

Generative Retrieval Generative retrieval models retrieve relevant items by generating sub/either the identifiers or entire sequences of the items. GENRE (Cao et al., 2021) retrieves a document by generating the titles with a constrained beam search. DSI (Tay et al., 2022) assigns a unique ID to each item in the corpus and retrieves the item by generating the ID of the most relevant document. SEAL (Bevilacqua et al., 2022) retrieves any span from any position in the corpus by using FM-Index. GMR (Lee et al., 2022) retrieves the most relevant item by generating the whole sequence. Though high performance, as generative retrieval models solely rely on the information stored in their parameter, the information capacity is limited and fixed. To overcome the limitation, we propose Nonparametric Decoding (Np Decoding) for generative retrieval models. By replacing the decoder vocab embeddings with nonparametric contextualized vocab embeddings, the model is able to utilize not only the parametric space but also the nonparametric space of contextualized embeddings.

Memory Augmented Models KNN-LM (Khandelwal et al., 2020), TRIME (Zhong et al., 2022), RAG (Lewis et al., 2020), and RETRO (Borgeaud et al., 2022) are memory augmented models which use both the parametric space of the model and the non-parametric space of the external memory. KNN-LM improves the LM performance by generating the next token through interpolation between the nearest neighbor distribution (distance in the contextualized embedding space) and the model vocab distribution only during the inference step. TRIME expands the work to use the objective also during the training step. RAG and RETRO first retrieve relevant texts with the retriever from the external memory and generate the output based on the retrieved texts. Moreover, concurrent work NPM (Min et al., 2022) proposes a nonparametric masked language model which operates over the nonparametric distribution of the external memory. Generative retrieval models with Nonparametric Decoding also utilize the external memory, but rather than considering it as an external source, it is incorporated with the model by utilizing the external memory as decoder vocab embeddings.

3 Nonparametric Decoding

Generative retrieval is the task of retrieving the most relevant retrieval target (e.g., title, passage, document identifier) by generating the target token-by-token when given an input query. The training objective of the generative retrieval model is to maximize

$$P((t_1, \cdots, t_n)|q) = \prod_{i=1}^{n} P(t_i|q, t_{<i}) \quad (1)$$

where t_* denotes the tokens of the retrieval target and q is the input query. Such an approach has shown high performance while using a low storage footprint (Cao et al., 2021; Tay et al., 2022; Bevilacqua et al., 2022; Lee et al., 2022). However, it has limitation in that the model depends *solely* on the information encoded in its own parameters. Thus, the performance is likely to be bounded by how much information can be stored in the model parameter (Tay et al., 2022; Roberts et al., 2020).

To address the limitation, we propose a new decoding method called Nonparametric Decoding (Np Decoding) for generative retrieval. To incorporate Np Decoding on the existing generative retrieval model, the only amendment is to use the frozen contextualized vocab embedding (external memory) rather than the vanilla vocab embedding as the decoder vocab embedding during each generation step (Figure 1). The embeddings are the output embeddings of an encoder when given a target sequence as input. Note that existing generative retrieval models such as GENRE and DSI utilize the pre-trained language model architecture as-is: vanilla vocab embedding as the decoder vocab embedding.

In Section 3.1, we show the key benefits of using Np Decoding over vanilla decoding. For Section 3.2 to Section 3.4, we show the details of base Np Decoding (BASE), and two variants (ASYNC, CONTRA). In Section 3.5, we describe how we reduce the number of contextualized token embeddings.

3.1 Key Benefits

Using Np Decoding has two key benefits over vanilla decoding. First, the generative retrieval model with Np Decoding can utilize not only the information encoded in its own parameters (parametric space) but also the surrounding information encoded in the contextualized vocab embeddings (nonparametric space) during each decoding step. Second, the generative retrieval model with Np Decoding has more expressive and fine-grained decoder vocab embedding space than that of the model with vanilla decoding. As in Figure 1, Np Decoding allows a single token to have multiple contextualized token embeddings for the decoder vocab embeddings (e.g., the same token "Cape" has two different contextualized embeddings) depending on the surrounding information of the token, whereas vanilla decoding allows only a single token embedding for a single token. Note that we do not save all possible token embeddings, but reduce the number of tokens to save without performance degradation by practical tactics (Section 3.5).

3.2 BASE Nonparametric Decoding

In this work, we propose three different Np Decoding (Base Nonparametric Decoding and two variants) which we name the three different Np Decoding based on the characteristics of the Contextualized Embedding Encoders (CE Encoder). CE Encoder is an encoder that outputs contextualized token embeddings when given a target sequence (e.g., title, document ID, passage) as input. The contextualized token embeddings are added to CE[2],

[2]Details of how we construct CE for different target sequences are in Section 4.3.

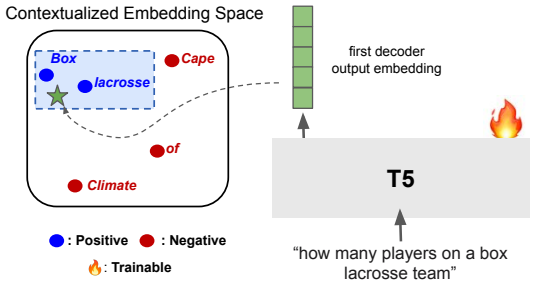

Figure 2: Token-level contrastive learning of CONTRA Np Decoding. Given a query ("how many players on a box lacrosse team") and target sequence ("Box lacrosse"), we train T5 on token-level contrastive learning where all tokens of the target sequence are the positive pairs and the rest of the tokens in CE are negative pairs.

the decoder vocab embedding matrix of generative retriever with Np Decoding. BASE Nonparametric Decoding (BASE) uses the most basic CE Encoder, the pre-trained T5 encoder as-is. CE is constructed once with the output embeddings of CE Encoder before the generative retrieval training step. Note that during the training step of the generative retrieval, CE Encoder is frozen (Figure 1).

3.3 ASYNC Nonparametric Decoding

Asynchronous Nonparametric Decoding (ASYNC) uses CE Encoder which is *asynchronously replaced* every N epoch by the encoder of generative retriever during the generative retrieval training step. By replacing CE Encoder periodically, ASYNC has more coherency between CE Encoder and the generative retriever than BASE. After every replacement (N epoch), we construct a new CE with the output embeddings of replaced CE Encoder and resume training the generative retriever. Note that during the generative retrieval training step, CE Encoder is frozen but simply replaced, and only generative retriever is trainable. We keep $N = 20$ for all experiments. See Appendix C.3 for details on how N affects the performance.

3.4 CONTRASTIVE Nonparametric Decoding

CONTRASTIVE Nonparametric Decoding (CONTRA) uses CE Encoder trained on *token-level contrastive learning*. The CE Encoder constructs CE, the nonparametric decoder vocab space of generative retrieval model with Np Decoding. The token-level contrastive learning (Equation 2) is performed as an intermediate step before training T5 on the generative retrieval task (Equation 1). Bi-encoder retrieval models with contrastive loss have shown high performance as the model learns to construct well-structured global embedding space and regularize the space to be uniform (Ni et al., 2021; Gao et al., 2021b; Gao and Callan, 2022; Izacard et al., 2022). In a similar way, CE Encoder with contrastive learning constructs a more meaningful dense vector space (non-parametric space of the generative retriever) than CE Encoder of BASE.

As in Figure 2, given a query, we train the first output embedding of the T5 decoder[3] with all tokens of the target sequence as positive pairs and the rest of the tokens in CE[4] as negative pairs. After training T5 with token-level contrastive learning, we construct the CE with its encoder as CE Encoder, and then further train the model on the generative retrieval task.

Step 1. Token-level Contrastive Learning Given a training dataset of pairs $\{(\boldsymbol{q}, \boldsymbol{t})\}$ where $\boldsymbol{q}$ is the query text, and $\boldsymbol{t}$ is the retrieval target (e.g., the title of the document to retrieve) composed of multiple tokens $\boldsymbol{t}_i$ ($1 \leq i \leq k$ where k is the length of the target), we split the training dataset into k separate pairs $\{(\boldsymbol{q}, \boldsymbol{t}_i)\}$ to construct the training dataset of query-token. With the query-token dataset, we train the first output token embedding from the T5 decoder to be close to all token embeddings in $\mathcal{T}^+$ when given query $\boldsymbol{q}$ as an input to generative retriever (Figure 2). $\mathcal{T}^+$ is a set of positive token embeddings[5] (tokens that make up one retrieval target), and $\mathcal{T}^-$ is the set of negative token embeddings[6] (all other token embeddings in CE). The objective is to minimize the contrastive loss:

$$L(\mathbf{q}, \mathbf{t}_1^+, \cdots, \mathbf{t}_{|\mathcal{T}^+|}^+, \mathbf{t}_1^-, \cdots, \mathbf{t}_{|\mathcal{T}^-|}^-) = -\log \frac{\sum_{\mathbf{t}^+ \in \mathcal{T}^+} e^{<\mathbf{q}, \mathbf{t}^+>}}{\sum_{\mathbf{t}^+ \in \mathcal{T}^+} e^{<\mathbf{q}, \mathbf{t}^+>} + \sum_{\mathbf{t}^- \in \mathcal{T}^-} e^{<\mathbf{q}, \mathbf{t}^->}} \quad (2)$$

where $\langle\ ,\ \rangle$ is the inner product value between the two embeddings. We also experiment with a contrastive loss having a single token per target as positive and in-batch negatives loss (Appendix A.1) where the contrastive loss with multiple tokens

[3] We use the embedding of decoder (Ni et al., 2021), not the encoder, to initialize generative retriever with both the encoder and the decoder trained on contrastive learning.

[4] As we freeze the token embeddings (CE) and only train the T5, calculating over entire embedding space is possible. CE used in the step is constructed with the output embeddings of the pre-trained T5 encoder model.

[5] $\mathcal{T}^+ = \{\mathbf{t}_1^+, \cdots, \mathbf{t}_k^+\}$ ($k = |\mathcal{T}^+|$)

[6] $\mathcal{T}^- = \{\mathbf{t}_1^-, \cdots, \mathbf{t}_{|\mathcal{T}^-|}^-\}$

(Equation 2) as positive shows the highest performance, which we hypothesize is because the positives have similar content information encoded.

Step 2. Generative Retrieval After training T5 with token-level contrastive learning, we use the trained encoder as CE Encoder and construct a new CE. We then further train the model on the generative retrieval task using the newly constructed CE as the decoder vocab embeddings[7].

3.5 Clustering

To construct CE, the decoder vocab embedding matrix of generative retriever, we first extract all contextualized embeddings of each target token with CE Encoder. As it requires a large storage footprint to save all the embeddings, we reduce the number of embeddings by using clustering and saving only the representative embeddings of each cluster. To be specific, we perform k-means clustering over the contextualized embeddings of the same token (which might have different surrounding contexts) and leave only the k centroid embeddings[8] as the decoder vocab embeddings of the token. We keep $k = 5$ for all experiments. When $k = 5$, it only requires 0.3% of storage footprint compared to when saving all contextualized token embedding. Also, it requires only 0.34GB more storage compared to the vanilla vocab embeddings ($k = 1$) which is marginal compared to the storage footprint to save the model parameters (3GB). See Appendix A.2 for details.

4 Experimental Setup

In section 4.1 and section 4.2, we describe the baselines and the datasets we used for experiments. In section 4.3, we show how we construct CE depending on which generative retriever we combined with. We experiment over 9 datasets. Results show that by simply replacing the decoding strategy, generative retrieval shows significantly higher performance. See Appendix B for more details about setups.

[7]The generative retrieval model is initialized with the T5 trained with token-level contrastive learning.

[8]When the number of extracted contextualized embeddings of a token is smaller than k, we do not perform k-means clustering but use its own contextualized embedding. Also, we use a single non-contextualized embedding for special tokens such as the EOS token or PAD token.

4.1 Baselines

BM25 (Robertson and Zaragoza, 2009) is a term-matching model relying on an efficient algorithm. DPR (Karpukhin et al., 2020) is a bi-encoder retrieval model which retrieves the most relevant document by performing a nearest neighbor search over dense vector space. Sentence-T5 (Ni et al., 2021) is similar to that of DPR but with T5 (Raffel et al., 2020) as the base model. MDR (Xiong et al., 2021b) is an extension of DPR to multi-hop datasets by iterating over a single query. More details about the baselines are in Appendix B.2. See Section 2 for descriptions of GENRE (Cao et al., 2021), DSI (Tay et al., 2022), and GMR (Lee et al., 2022).

4.2 Datasets & Evaluation Metrics

We use 9 datasets with various characteristics: FEVER (Thorne et al., 2018), AY2 (Hoffart et al., 2011), TREX (ElSahar et al., 2018), zsRE (Levy et al., 2017), NQ (Kwiatkowski et al., 2019), TQA (Joshi et al., 2017), WOW (Dinan et al., 2019), NQ-320k (Tay et al., 2022), and HotpotQA (Yang et al., 2018). For all datasets except for NQ-320k and HotpotQA, we use dataset and corpus from KILT (Petroni et al., 2021). To compare with DSI (Tay et al., 2022), we experiment over NQ-320k, a restricted setting from the official NQ dataset; it uses about 4% of Wikipedia as the corpus set. Note that the NQ of the KILT version and the official version is different (Details are in Petroni et al. (2021)). HotpotQA (Yang et al., 2018) is an open-domain multi-hop question-answering dataset that needs two Wikipedia pages to answer the question. For HotpotQA we used the official version of the dataset and the corpus.

We evaluate all results of the KILT version with R-precision, a metric widely used to evaluate retrieval performance in KILT. It is calculated as $\frac{r}{R}$ where R is the number of Wikipedia documents in each provenance set, and r is the number of related documents among the top-R retrieved documents. The results of NQ-320k and HotpotQA are evaluated using Hits@N (N={1, 10}), which shows the proportion of the correct documents ranked in the top N predictions.

4.3 Details of Constructing CE

The choice of which output embeddings of CE Encoder to save when constructing CE depends on the target sequence of the generative retrieval model.

Model	FEVER	AY2	TREX	zsRE	NQ	TQA	WOW	Avg
BM25	38.2	1.4	57.4	66.3	23.4	25.2	23.9	33.7
DPR	73.0	44.6	72.9	94.1	**60.1**	63.9	36.5	63.6
GENRE*	70.3	**75.6**	73.9	97.0	51.8	65.0	59.2	70.4
G*-BASE	73.3	73.0	79.2	99.1	59.0	68.2	61.1	73.3
G*-ASYNC	74.7	74.6	78.2	98.9	59.2	68.4	61.9	73.7
G*-CONTRA	**77.1**	75.5	**81.3**	**99.2**	59.8	**68.6**	**62.4**	**74.8**

Table 1: R-precision(%) for document retrieval task on test dataset when trained with each dataset (KILT version). G*-BASE, G*-ASYNC, and G*-CONTRA are the results when adding Np Decoding on GENRE*. The best and second best of each dataset in **bold** and underline respectively.

	Hits@1	Hits@10
BM25	11.6	34.4
Sentence-T5	22.4	63.3
DSI_{Naive}	13.3	33.6
$DSI_{Semantic}$	35.6	62.6
DSI_{Naive}-BASE	58.7	73.1
DSI_{Naive}-CONTRA	**60.4**	**75.8**
GENRE*	53.7	64.7
GENRE*-BASE	62.2	78.8
GENRE*-CONTRA	**63.4**	**81.1**

Table 2: Hits@1 and @10 in NQ-320k. Results of BM25, Sentence-T5, DSI_{Naive}, and $DSI_{Semantic}$ are from Tay et al. (2022). Best of each section in **bold**.

In this work, we focused on applying Np Decoding to generative retrieval models that use representative words as the target sequence (DSI and GENRE) and leave extending the work to generative retrieval models that need the whole sequence as the target sequence (GMR and SEAL) as future work.

Np Decoding on GENRE* As the target sequence of GENRE*[9] is the title of the most relevant document, we construct CE with the output embeddings of document titles. To additionally encode the content of the document in the title embeddings, we input the title and the document content[10] into CE Encoder and save only the output embeddings of the title when constructing CE.

Np Decoding on DSI As the target sequence of DSI is the document ID of the most relevant document, we construct CE with the document ID[11]. As in GENRE*-Np Decoding, we input the document ID with the document content as the input of CE Encoder.

5 Experimental Results

In this section, we demonstrate the benefits of using Np Decoding in generative retrieval models by comparing the performance of existing generative retrieval models (DSI, GENRE*) with and without the method in document retrieval tasks.

[9]GENRE* is GENRE trained with T5-large. We used T5 as a base model for a fair comparison with other base models. See Appendix C.2 for a performance comparison between GENRE* (T5-large) and GENRE (Bart-large).

[10]We use the first five paragraphs of the document to the maximum of 512 tokens as the document content due to limited input length.

[11]As the document ID of official DSI is not released, we assign the document ID with arbitrary unique integers (naively structured identifiers, DSI_{Naive}). DSI_{Naive} with our document ID shows 12.5 and 22.4 for Hits@1 and Hits@10.

5.1 Vanilla Decoding vs. Np Decoding

Table 1 shows that using Np Decoding improves task-specific performance on various datasets, with an average of 4.4% improvement. The improvement is especially significant on datasets with a large number of training examples (FEVER, TREX), which we assume is because the model can learn more about the new vocab embeddings (CE) during the training step.

Table 2 shows the results of NQ-320k when applying Np Decoding on DSI and GENRE*. The two models differ in that the retrieval target of DSI is document ID and that of GENRE* is document title. Np Decoding enhances DSI and GENRE* by 24.8% and 9.7% in Hits@1. The results suggest that applying Np Decoding is especially helpful for cases where the vanilla vocab embeddings of the target sequence have less information; improvement is higher in DSI than in GENRE*. As the document ID is constructed with arbitrary unique integers, the vanilla vocab embeddings of document ID contain less semantic information and have not seen the relationship between the document and the ID during the pretraining step. In contrast, as the title uses vanilla vocab embeddings of natural language, the embeddings would contain more information compared to that of the document ID.

Table 3 shows that using Np Decoding also improves performance on HotpotQA, the multi-hop retrieval dataset; GENRE*-CONTRA shows a 7%-increase in performance compared to GENRE* in Recall@2. Also, compared to GMR, as GENRE*-CONTRA is able to capture the entire context information with just title generation by using the contextualized embeddings, it shows higher performance and faster inference speed. Furthermore, when compared to a multi-hop bi-encoder model MDR-, a variant of MDR without advanced tech-

niques like linked negatives, memory bank, and shared encoder, GENRE*-CONTRA shows higher performance, whereas lower performance compared to MDR, a model with all the techniques are applied. We expect that applying such techniques to generative retrieval models would also be helpful and leave it as a future work. More details about the results and how we extend GENRE* in the multi-hop setting are in Appendix C.1.

5.2 Benefits of Nonparametric Decoding

We found three major benefits of using Np Decoding over vanilla decoding for generative retrieval.

(1) Parameter-Efficient GENRE*-BASE trained with T5-base (54.0 and 66.4) shows higher performance in NQ and TQA compared to GENRE* trained with T5-large (51.8 and 65.0) while T5-large has 3.5 times more parameters than T5-base. Also, DSI-BASE (58.7) shows about 1.5× higher performance in NQ-320k Hits@1 compared to DSI T5-XXL with Semantic String Docid (the best-performing model in Tay et al. (2022)) (40.4), which has 14 times more parameters than DSI-BASE, demonstrating that a retrieval model with Np Decoding is more parameter efficient.

(2) Data-Efficient GENRE*-CONTRA trained on NQ and TQA together has similar performance to GENRE trained on the entire KILT dataset together, despite having only 5% as many training examples. To be specific, when evaluated on R-precision, GENRE*-CONTRA performs 60.3/68.9, and GENRE performs 60.3/69.2 in NQ/TQA, respectively (Table 7 in Appendix C.2). Such results demonstrate that using Np Decoding is advantageous in the low-resource setting as it can utilize the information in the non-parametric space.

(3) Robust to Zero-Shot Table 5 shows that GENRE*-BASE is stronger than GENRE* in the KILT zero-shot setting where both models are trained on NQ and TQA together and are evaluated on the other 9 datasets in KILT that are not used during the training step. GENRE*-BASE shows an average of 3% improvement over GENRE*. GENRE*-BASE is able to generalize well to out of domains as it does not rely only on the information encoded in the parametric space but also utilizes the nonparametric space of CE which is shared across all domains; Np Decoding makes the generative retrieval model more robust in the zero-shot setting.

Table 3: Recall rate of HotpotQA official full-wiki dev set. Results of DPR, MDR-, and MDR are from Xiong et al. (2021b) and results of GMR are from Lee et al. (2022). MDR- indicates a variant of MDR without linked negatives, memory bank, and shared encoder. Best among each method in **bold**.

Method	Model	Recall@2	Recall@10
Bi-Encoder	DPR	25.2	45.4
	MDR-	59.9	70.6
	MDR	**65.9**	**77.5**
Generative	GMR	57.7	58.8
	GENRE*	56.1	58.4
	GENRE*-BASE	61.9	65.3
	GENRE*-CONTRA	**63.1**	**66.8**

(4) Robust to Low Lexical Overlap To evaluate whether the model leverages the information encoded in CE when using Np Decoding, we test the performance of GENRE*-BASE and GENRE* on queries that are likely to require utilizing information from document content (queries with low lexical overlap with the target sequence) in order to find the answer. We divide the queries in the NQ dev set into low- and high-overlap sets using the TF-IDF score. GENRE* and GENRE*-BASE both show relatively high performance on queries in the high-overlap set compared to the low-overlap set as it is easier to infer the correct retrieval target from the query alone even if the model does not know the document content. However, GENRE*-BASE shows about 7% higher performance on the low-overlap set and 5% higher performance on the high-overlap than GENRE*. This shows that GENRE* with Np Decoding (GENRE*-BASE) is robust on queries in the low-overlap set by utilizing the information encoded in CE. (More details in Appendix C.2.)

5.3 What is Well-Constructed Contextualized Embedding Matrix (CE)?

We found that the choice of CE plays an important role in the performance when applying Np Decoding. We analyzed four factors that are especially important to create a well-constructed contextualized embedding matrix (CE). See Appendix C.3 for various analyses of contextualized token embeddings and more details of each subsection.

(1) Coherency between Generative Retrieval and CE Encoder Table 1 shows that ASYNC, which replaces CE using the encoder of generative retriever every N epochs, tends to show higher performance than BASE Np Decoding, which uses fixed CE. Also, updating CE more frequently

Table 4: Top-3 prediction results of GENRE*-BASE, GENRE*-BASE-Short, and GENRE* on NQ dev set in KILT. Highlights on the correct target sequence.

Query	Prediction Results	
what do the 3 dots mean in math	**GENRE*-BASE**	Therefore sign, Infinity symbol, Equation
	GENRE*-BASE-Short	Slashed zero, Homo sapiens, Equation
	GENRE*	Ellipsis, Infinity symbol, Homo sapiens
rizal finished all the chapters of the novel noli me tangere in	**GENRE*-BASE**	Noli Me Tángere (novel), Noli Me Tangere (opera), Noli Me Tangere (Bernini)
	GENRE*-BASE-Short	Noli me tangere, Noli Me Tángere (novel), Noli Me Tangere (opera)
	GENRE*	Noli me tangere, Non è l'inferno, Noli Me Tangere (opera)

Table 5: R-precision(%) for the test sets of document retrieval tasks on datasets in KILT. Both GENRE* and GENRE*-BASE are trained with NQ + TQA; other datasets are not seen during the training time. Best in **Bold.**

	In-Domain Datasets		Out-of-Domain Datasets (Inference Only, Zero-Shot)									
	NQ	TQA	FEVER	AY2	WnWi	WnCw	T-REX	zsRE	HoPo	ELI5	WoW	OoD Avg
GENRE*	52.7	64.8	64.2	9.1	2.8	3.4	53.9	76.1	34.3	11.2	48.9	33.8
GENRE*-BASE	**59.4**	**68.7**	**67.0**	**10.3**	**5.4**	**7.8**	**59.1**	**79.2**	**37.5**	**12.5**	**51.7**	**36.7**

(smaller N) leads to better performance. Such results suggest that having high coherency between generative retriever and CE Encoder improves the performance. However, as it needs extra cost to construct CE for each update, there is a tradeoff between the computation overhead and the performance.

(2) Contrastive Learning Table 1,2,3 show that CONTRA shows consistently higher performance than BASE. The results indicate that CE Encoder trained on contrastive loss tends to construct better CE by leveraging the benefits of contrastive learning of constructing well-structured overall embedding space and regularizing the space to be uniform (Ni et al., 2021; Gao et al., 2021a,b; Izacard et al., 2022). When we calculate $L_{\text{uniformity}}$, a metric which checks how well the embedding space is constructed (Wang and Isola, 2020), CONTRA (-19.7) shows a lower number than BASE (-18.2) where the lower the better.

(3) Contextualized Embeddings Size CE contains multiple contextualized embeddings for each token where the number of embeddings per token is controlled by the clustering method. In Figure 3, the performance shows the highest performance when using a maximum of five contextualized embeddings per token, and using a number higher or lower than five tends to decrease the performance for both NQ and TQA. This suggests that having too many vocab embeddings can be distracting while having too few can be not representative enough; the number of clusters should be neither too few nor too many.

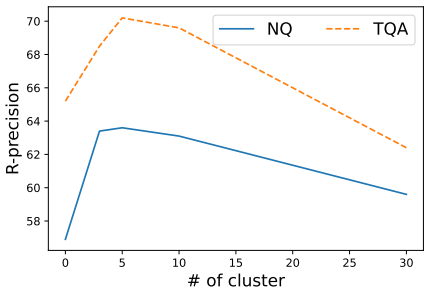

Figure 3: Effect of the maximum number of clusters (number of contextualized vocab embeddings) for each token on the performance of GENRE*-BASE. The results when the number of clusters is zero are the results of GENRE*.

(4) Longer Context We compare the results between BASE-Short and BASE where BASE-Short is a variant of BASE where CE is constructed using shorter context (*only the title* without the document content) as input to CE Encoder. While both BASE and BASE-Short use CE as decoder vocab embeddings, the contextualized vocab embeddings of BASE-Short contain less contextual information compared to those of BASE due to shorter context input to CE Encoder. Therefore, as shown in Table 4, BASE-Short performs poorly in cases where the document content is necessary for successful retrieval. Additionally, BASE-Short has lower R-precision in both NQ and TQA (58.4% and 68.2%) compared to BASE (59.4% and 68.7%), indicating

a correlation between performance and the amount of contextual information in CE (non-parametric space).

6 Conclusion

In this paper, we propose Nonparametric Decoding (Np Decoding), a new decoding method that can be applied to canonical generative retrieval models by simply replacing the decoder vocab embeddings from vanilla vocab embeddings to non-parametric contextualized vocab embeddings (output embeddings of an encoder). This way, the generative retrieval does not rely solely on the information encoded in its own model parameters but can also utilize the information encoded in the contextualized embeddings. Using Np Decoding in generative retrieval significantly improves the performance, achieves higher data and parameter efficiency, and shows more robustness in the zero-shot setting. In future work, we plan to apply Np Decoding to various other tasks beyond task retrieval.

7 Limitations

Np Decoding uses k-means clustering to reduce the number of contextualized embeddings, the performance varies by how the contextualized embeddings are clustered. As the process is relatively inconsistent, reducing the number with other methods would make the model performance more consistent. Also, as it is not trivial to add new contextualized token embeddings on top of pre-constructed CE due to the clustering step, we did not perform on dynamic corpus setup where new items are added or updated.

Np Decoding is applicable to all generative retrieval models including GMR or SEAL which needs all token embeddings, however, we focused on generative retrieval models with representative output as the retrieval target in this work. Also, while it is a general approach applicable to all encoder-decoder models, we focused on applying the method to T5.

Acknowledgements

We thank Seonghyeon Ye, Joel Jang, and Sejune Joo for constructive feedback. This work was partly supported by Kakao Brain grant (2021, Memory-Augmented Language Model, 80%) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00264, Comprehensive Video Understanding and Generation with Knowledge-based Deep Logic Neural Network, 20%).

References

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. In *NeurIPS*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, T. W. Hennigan, Saffron Huang, Lorenzo Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and L. Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *ICML*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *ICLR*.

W. Bruce Croft and John D. Lafferty. 2010. Language modeling for information retrieval. In *The Springer International Series on Information Retrieval*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*.

Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Paslaru Bontas Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *LREC*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *ArXiv*.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *ACL*.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *NAACL*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP*.

Zhaochen Guo and Denilson Barbosa. 2014. Robust entity linking via random walks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *TMLR*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *ICLR*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *TACL*.

John D. Lafferty and ChengXiang Zhai. 2003. Probabilistic relevance models based on document and query generation.

Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative multi-hop retrieval. *EMNLP*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *TACL*.

Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen tau Yih, Barlas Oğuz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-task retrieval for knowledge-intensive tasks. In *ACL*.

Sewon Min, Weijia Shi, Mike Lewis, Xilun Chen, Wen tau Yih, Hanna Hajishirzi, and Luke Zettlemoyer. 2022. Nonparametric masked language modeling. *ArXiv*.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *CoRR*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *NAACL*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *EMNLP*.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *NeurIPS*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *NACCL*.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *PMLR*.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A neural corpus indexer for document retrieval. In *NeurIPS*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP*.

Ledell Yu Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *EMNLP*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021a. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *ICLR*.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021b. Answering complex open-domain questions with multi-hop dense retrieval. In *ICLR*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Zexuan Zhong, Tao Lei, and Danqi Chen. 2022. Training language models with memory augmentation.

A Nonparametric Decoding

A.1 Different Types of Contrastive Loss for CONTRA Np Decoding

We experiment with three different types of contrastive loss when training CONTRA. In this section, we show the losses and how the results differ by each loss.

Given a training dataset of pairs $\{(\boldsymbol{q}, \boldsymbol{t})\}$ where $\boldsymbol{q}$ is the query text, and $\boldsymbol{t}$ is the retrieval target (title of the document to retrieve) composed of multiple tokens $\boldsymbol{t}_i$ ($1 \leq i \leq k$ where k is the length of the target), we split all tokens into k separate pairs $\{(\boldsymbol{q}, \boldsymbol{t}_i)\}$ to construct the training dataset of query-token. The three loss differs in what the model considers as a negative set and a positive set.

Loss 1: Neg: In-Batch Negatives / Pos: Single Token Embedding With the query-token dataset, we train generative retriever's first output token representation from the decoder to be close to all $\mathbf{t}^+ \in \{\mathbf{t}_1, \cdots, \mathbf{t}_k\}$ (embedding of any token in the retrieval target $\boldsymbol{t}$) given the query $\boldsymbol{q}$ as an input to generative retriever. The objective is to minimize the contrastive loss to make the query text embedding $\mathbf{q}$ be closer to positive token embedding $\mathbf{t}^+$:

$$L(\mathbf{q}, \mathbf{t}^+, \mathbf{t}_1^-, \cdots, \mathbf{t}_{|\mathcal{T}^-|}^-) \tag{3}$$

$$= -\log \frac{e^{<\mathbf{q},\mathbf{t}^+>}}{e^{<\mathbf{q},\mathbf{t}^+>} + \sum_{\mathbf{t}^- \in \mathcal{T}^-} e^{<\mathbf{q},\mathbf{t}^->}} \tag{4}$$

where $\langle\ ,\ \rangle$ is the inner product value between the two embeddings, and $\mathcal{T}^- = \{\mathbf{t}_1^-, \cdots, \mathbf{t}_{|\mathcal{T}^-|}^-\}$ is the set of negative token embeddings, which are other token embeddings in the training batch that are not paired with $\boldsymbol{q}$ (in-batch negatives (Karpukhin et al., 2020)).

Loss 2: Neg: Contextualized Embedding Matrix / Pos: Single Token Embedding The loss differs from the upper loss in that it considers *all* embeddings in contextualized embedding matrix except the single positive embedding as negative rather than performing the in-batch negatives which consider the subset of contextualized embedding matrix as negatives. The equation is same as Equation 3, but elements in $\mathcal{T}^-$ are *all* other token embeddings in contextualized embedding matrix.

Loss 3: Neg: Contextualized Embedding Matrix / Pos: Multiple Token Embedding The loss differs from the upper loss in that it considers *all* token embeddings in the title as positive embeddings; for each query $\boldsymbol{q}$, there are more than one positive contextualized token embeddings.

With the query-token dataset, where $\mathcal{T}^+ = \{\mathbf{t}_1^+, \cdots, \mathbf{t}_k^+\}$, set of positive token embeddings, we train generative retriever's first output token representation from the decoder to be close to all token embeddings in $\mathcal{T}^+$ given the query $\boldsymbol{q}$ as an input to generative retriever. The objective is to minimize the contrastive loss to make the query text embedding $\mathbf{q}$ be closer to all positive token embedding in $\mathcal{T}^+$:

$$L(\mathbf{q}, \mathbf{t}_1^+, \cdots, \mathbf{t}_{|\mathcal{T}^+|}^+, \mathbf{t}_1^-, \cdots, \mathbf{t}_{|\mathcal{T}^-|}^-) \tag{5}$$

$$= -\log \frac{\sum_{\mathbf{t}^+ \in \mathcal{T}^+} e^{<\mathbf{q},\mathbf{t}^+>}}{\sum_{\mathbf{t}^+ \in \mathcal{T}^+} e^{<\mathbf{q},\mathbf{t}^+>} + \sum_{\mathbf{t}^- \in \mathcal{T}^-} e^{<\mathbf{q},\mathbf{t}^->}} \tag{6}$$

where $\langle\ ,\ \rangle$ is the inner product value between the two embeddings, and $\mathcal{T}^- = \{\mathbf{t}_1^-, \cdots, \mathbf{t}_{|\mathcal{T}^-|}^-\}$ is the set of negative token embeddings, which are *all* other token embeddings in contextualized embedding matrix.

A.2 Clustering

Example When a token "the" appears in the corpus 100 times, 100 different contextualized embeddings of "the" are extracted by the encoder model at first. Then, we perform k-means clustering on the 100 contextualized embeddings to cluster them into at most k clusters and save all centroid embeddings. We leave only the k centroid embeddings as

the decoder vocab embeddings of the token "the" and assign a new decoder token ID for each contextualized embedding by the cluster it belongs to. By repeating the process over all the tokens, each token has a number of contextualized embeddings less or equal to k. As there are multiple contextualized token embeddings for a single token, we replace the ground-truth target token IDs with the newly constructed decoder token IDs to specify which contextualized token embedding the ground-truth target token ID is referring to.

Storage Footprint We analyzed how much storage footprint can be saved through the clustering method. When we use the KILT version Wikipedia corpus and titles as the retrieval target, about 37M token embeddings need to be stored. If the maximum number of token embeddings per token is set to 5 ($k = 5$), only about 117K token embeddings need to be stored. Also, vanilla vocab embeddings of T5 ($k = 1$) use about 32K token embeddings. Therefore, when $k = 5$, it only needs 0.3% of the storage footprint compared to when storing all token embeddings of the title and about 3.7 times more storage compared to the vanilla vocab embeddings. When $k = 5$, it needs 0.47GB of storage footprint to save all the vocab embeddings, whereas the vanilla vocab embeddings ($k = 1$) need 0.13GB. The increase in the storage footprint of vocab embeddings (0.34GB) is marginal compared to the storage footprint to save the model parameters (3GB).

B Experimental Setup

B.1 GENRE* and GENRE* with Np Decoding

We train all models using a pre-trained T5-large (Raffel et al., 2020) checkpoint from Wolf et al. (2020) as the initial checkpoint (770M parameters). GENRE* and all generative retrieval models with Np Decoding are trained with the same hyperparameter setting for a fair comparison. The training was done on 8 32GB V100 GPUs or a similar device. We train using Adafactor with a learning rate 1e-4[12] with a linear warm-up for the first 10% of training and then linear decay with batch size 512 till a maximum of 150 epochs with early stopping. All results are from a single run.

[12]We also tried with a learning rate of 1e-3, a commonly used learning rate, but le-4 shows consistently higher performance.

B.2 BM25 & DPR

To match the setting (dataset) similar to other baseline models, we train DPR (Karpukhin et al., 2020) in a document retrieval task. Unlike Maillard et al. (2021), which performs document retrieval tasks by training the model on passage-level tasks and considers the retrieval successful if it retrieves the passage in the target document, we train DPR on document-level tasks so that it retrieves the document itself. We consider the first five paragraphs as the content and train the model so that the query embedding gets close to not the paragraph embedding but the document embedding. We use only the first five paragraphs of each document due to the limit in input length, and to keep it the same as the information used when dumping CE by CE Encoder. The number of the corpus in the document retrieval tasks is the same as the number of pages in the KILT dataset. For BM25, we use pyserini (Lin et al., 2021) where the corpus is the same as in DPR. All results are from a single run.

B.3 Datasets

For zero-shot evaluation, we also evaluated over WnWi (Guo and Barbosa, 2014) and ELI5 (Fan et al., 2019).

B.4 Prefix Tree

We perform a constrained beam search with prefix tree (Cao et al., 2021) during the inference step to assure that all generated sequences are in the corpus. The prefix tree is constructed with the tokenization result of the corpus, and we perform a constrained beam search by masking out the tokens that do not create a sub-string of the text in the corpus. We find the next tokens from the top-k of the unmasked ones. While *token ID* was used as the node of the prefix tree in previous works since each token was mapped to a unique token ID, we construct a prefix tree with the *text* of the token as the node, because CE contains multiple token IDs for a single token. Therefore, rather than unmasking only a single token ID, we unmask all token IDs that correspond to the text in order to unmask a token. We keep the beam size to 10 for all experiments following Cao et al. (2021).

C Experimental Results

C.1 Multi-hop Dataset

GENRE* in multi-hop setting During the inference step of GENRE*, in the first hop, GENRE*

Table 6: HotpotQA bridge vs. comparison (Top2)

	Bridge	Comparison
MDR	**58.7**	94.8
GMR	47.9	96.4
GENRE*	47.2	91.4
GENRE*-BASE	53.1	96.6
GENRE*-CONTRA	54.9	**96.9**

retrieves the title of the most relevant document (T1) when given a query, and in the second hop, GENRE* retrieves the title of the most relevant document (T2) when given the query, T1, and the context of T1 as input to the model.

Bridge vs. Comparison Questions HotpotQA contains both the bridge and the comparison questions; bridge questions are those that need to infer the missing intermediate entity from the document content of the first hop, and comparison questions are those with the two entities mentioned simultaneously. We analyzed the performance of MDR, GMR, GENRE*, and GENRE* with Np Decoding (GENRE*-BASE, GENRE*-CONTRA) by dividing the performance into bridge and comparison questions. GENRE* with Np Decoding shows the highest performance in comparison questions among all models, and the highest performance in bridge questions among the generative retrieval models.

C.2 Benefits of Nonparametric Decoding

Multitask Training Results in Table 7 show that GENRE*-CONTRA outperforms GENRE* by 6% in single-task which demonstrates the effectiveness of Np Decoding. For both cases where the model is trained over a single dataset and over NQ and TQA together (NQ+TQA), GENRE* with Np Decoding shows higher performance over GENRE*. Note that due to limited available resources, we did not train GENRE* with Np Decoding on the full KILT dataset (ALL KILT) as in GENRE[13], DPR, or SEAL. However, CONTRA trained on less than 5% of the training dataset from the full KILT dataset shows higher or comparable performance to those models.

Robust to Low Lexical Overlap We first run TF-IDF over all the queries of the NQ dev set in KILT and divide the queries into two sets: low-overlap[14] and high-overlap[15]. Low-overlap is a set of queries with a TF-IDF score lower than average, and high-overlap is the rest of the queries.

Generative retrieval with Np Decoding shows especially strong performance on queries in the low-overlap set; queries that in most cases need the context information unless the model saw the information during the training step. We check four sets:

1. GENRE+/BASE +: queries where both BASE and GENRE* successfully retrieved
2. GENRE+/BASE-: queries where GENRE* successfully retrieved and BASE failed
3. GENRE-/BASE +: queries where GENRE* failed and BASE succeed
4. GENRE-/BASE-: queries where GENRE* and BASE both failed.

Figure 4 and Figure 5 show the low-rate (blue) and high-rate (red). Low-rate of each case is calculated as $\frac{\{Q \cap L\}}{Q}$, where Q is a set of queries in each case and L is a set of queries in a low-overlap set. High-rate of each case is calculated as $\frac{\{Q \cap H\}}{Q}$, where H is a set of queries in a high-overlap set.

For both figures, GENRE-/BASE + shows a higher number in low-rate, which indicates that BASE tend to successfully predict queries in the low-overlap set compared to GENRE*. Also, for both figures, GENRE+/BASE + shows a high number in low-rate and GENRE-/BASE- shows a high number of high-rate, which indicates that queries in the high-overlap set tend to be easy questions for both GENRE and BASE whereas queries in the low-overlap set are difficult for both models.

Also, Table 8 shows samples of the top-5 prediction results of BASE and GENRE* where BASE successfully retrieved the correct item and GENRE* failed. Moreover, Table 9 shows the performance of GENRE and GENRE*-BASE for low-overlap and high-overlap sets. The results suggest that BASE is robust on queries in the low-overlap set compared to GENRE*.

C.3 What is Well-Constructed Contextualized Embedding Matrix (CE)?

In this section, we analyze Np Decoding with GENRE* so we skip the model name.

[13] GENRE uses 128 V100 GPUs with 32GB of memory for about 33 hours.

[14] e.g., Q: During which season does cape town receive rainfall / Target Document: Climate of South Africa

[15] e.g., Q: where was the *world economic forum* held this year / Target Document: World Economic Forum

Training Dataset	Single ($\leq$ 3%)		NQ+TQA ($\leq$ 5%)		NQ+TQA+HotpotQA+ELI5 ($\leq$ 16%)				All KILT (100%)	
Model	NQ	TQA	NQ	TQA	NQ	TQA	ELI5	HotpotQA	NQ	TQA
GENRE	-	-	-	-	58.3	**69.6**	13.2	40.3	60.3	**69.2**
GENRE*	51.8	65.0	52.7	64.8	54.2	67.8	13.8	43.5	-	-
GENRE*-BASE	59.0	68.2	59.4	68.7	59.3	68.9	14.2	44.9	-	
GENRE*-ASYNC	59.2	68.4	59.8	68.7	60.1	69.1	14.0	46.3	-	-
GENRE*-CONTRA	59.8	**68.6**	**60.3**	**68.9**	**60.7**	68.6	**14.9**	**47.0**	-	-
BM25	23.4†	25.2†	23.4†	25.2†	23.4†	25.2†	5.3†	38.4†	23.4†	25.2†
DPR	**60.1**†	63.9†	59.5†	62.9†	58.0†	63.2†	10.6†	39.3†	59.4	61.5
SEAL	-	-	-	-	-	-	-	-	**63.2**	68.4

Table 7: R-precision(%) for document retrieval task on NQ and TQA test dataset (KILT version). Results except for GENRE* and GENRE* with Np Decoding (GENRE*-BASE, GENRE*-ASYNC, and GENRE*-CONTRA) are from the KILT leaderboard. The column of the table is divided by how many training datasets are used. Numbers in the bracket are the rate of the number of training datasets over the number of training datasets when using all KILT datasets. Results of GENRE and SEAL are from Cao et al. (2021) and Bevilacqua et al. (2022), respectively. Results with † in BM25 and DPR are trained in the same setting as GENRE* with Np Decoding (Appendix B.2). Best in **bold.**

Table 8: Top-3 prediction result of BASE, and GENRE*

Query	**Prediction Result**
what do the 3 dots mean in math	**BASE** Therefore sign, Infinity symbol, Equation
	GENRE Ellipsis, Infinity symbol, Homo sapiens
what does the pearl symbolize in the bible	**BASE** Parable of the Pearl, Mitzvah, Pearl of Wisdom
	GENRE Pearl of Great Price, Perlin, Promised Land
does archie end up with betty or veronica in riverdale	**BASE** Archie Marries Veronica/Archies Marries Betty, List of Riverdale characters, Archie Buchanan
	GENRE Riverdale (2017 TV series), List of Riverdale characters, Archie Mitchell
actor who plays dr avery on grey's anatomy	**BASE** Jesse Williams (actor), Jesse Williams, Jesse Spencer
	GENRE Marc Alaimo, Patrick Warburton, Jeffrey Dean Morgan
when did equus first appear in fossil record	**BASE** Evolution of the horse, Equis, Eurydice
	GENRE Equidae, Equis, Equinox
who decides the number of judges in the high court	**BASE** Indian High Courts Act 1861, High Court of Australia, Supreme Court of India
	GENRE Supreme Court of the United Kingdom, Supreme Court of India, High Court of Australia
when's the last time the philadelphia eagles played the new england patriots	**BASE** Super Bowl XXXIX, New England Patriots, Super Bowl XXXVIII
	GENRE New England Patriots, Philadelphia Eagles, History of the Philadelphia Eagles
rizal finished all the chapters of the novel noli me tangere in	**BASE** Noli Me Tángere (novel), Noli Me Tangere (opera), Noli Me Tangere (Bernini)
	GENRE Noli me tangere, Non è l'inferno, Noli Me Tangere (opera)
during which season does cape town receive rainfall	**BASE** Climate of South Africa, City of Cape Town, Cape Town water crisis
	GENRE Cape Town, City of Cape Town, Cape Town water crisis

(1) Having High Coherency between Generative Retrieval and CE Encoder We analyzed how the performance changes according to how often the replacement of CE Encoder by the encoder of the generative retrieval occurs (replacement for every N epoch) with ASYNC-Short. When comparing the performance with $N = \{10, 20, 50\}$, ASYNC-Short shows the highest performance at $N = 10$, and the performance tends to deteriorate as N becomes larger. Also, all ASYNC-Short show higher performance than BASE-Short, which uses CE with no replacement during training (N = max training epoch). Results show that although the model requires high computation cost and longer training time as N gets smaller, it is important to have high coherency between the contextualized

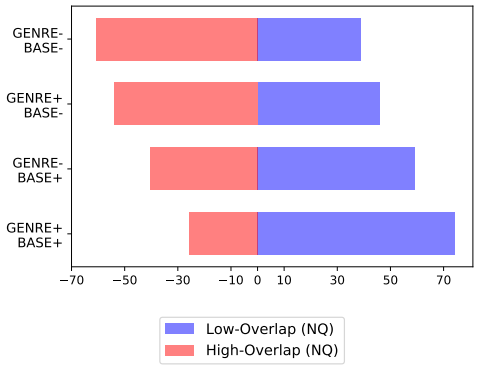

Figure 4: Red bar indicates the high rate and the blue bar indicates the low rate. The rate is measured by NQ dev set in KILT. Details about high-rate and low-rate is in Appendix C.2.

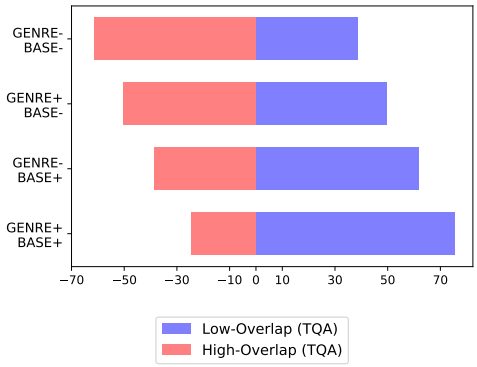

Figure 5: Red bar indicates the high rate and the blue bar indicates the low rate. The rate is measured by TQA dev set in KILT. Details about high-rate and low-rate is in Appendix C.2.

Table 9: R-precision(%) for the document retrieval task on NQ dev dataset in KILT. See details about Low and High Overlap in Appendix C.2.

Overlap	GENRE*	GENRE*-BASE-Short	GENRE*-BASE
Low	45.8	51.6	52.7
High	71.3	75.3	75.8
Total	58.3	63.2	64.0

Table 10: R-precision(%) for the document retrieval task on NQ and TQA test dataset in KILT. We compare the results of GENRE*, BASE-Short, and BASE where the models are trained with NQ+TQA. The results show the importance of extracting contextualized embeddings with not only the title but also the corresponding document content.

	GENRE*	GENRE*-BASE-Short	GENRE*-BASE
NQ	52.7	58.4	**59.4**
Trivia	64.8	68.2	**68.7**

embeddings (output embeddings of CE Encoder) and generative retriever by frequent replacement.

(2) Training CE with Contrastive Learning Appendix A.1 shows the details of three different contrastive losses that we experiment over when training CONTRA Np Decoding. Table 11 show the performance of CONTRA with different contrastive loss, which differs by what is considered as the positive pair and the negative pair. *Multiple Token Emb* considers all token embeddings in the same target sequence as positive pairs, and *Single Token Emb* considers all token embeddings separately thus only one of the token embedding from the title token embeddings is considered as positive pair. *In-Batch Negatives* considers all embeddings in a batch except for the positive embedding as negative pairs, and *Contextualized Embedding Matrix* considers all embeddings in the contextualized embedding matrix (a matrix constructed with the contextualized token embeddings) except for the positive embeddings as negative pairs.

The model trained on contrastive loss with multiple token embeddings as positive pairs, and all other embeddings in contextualized embedding matrix as negative pairs (Loss3) show the highest performance. The model trained on the same negative but with a single token embedding as positive (Loss2) shows the lowest performance. The model with single token embedding as positive and in-batch negatives as negative pairs (Loss1) shows the performance in-between.

As in Xiong et al. (2021a), the model with Loss2 and Loss3 has the benefits of looking at the global

Table 11: R-precision(%) for the document retrieval task on NQ and TQA test dataset in KILT. See Appendix C.3 for details about how the loss term differs. The loss term is used while training CONTRA in contrastive learning (step 1 of training CONTRA).

Positive	Negative	NQ	TQA
Single Token Emb	In-Batch Negatives	60.0	**68.9**
Single Token Emb	Contextualized Embedding Matrix	58.9	68.4
Multiple Token Emb	Contextualized Embedding Matrix	**60.3**	**68.9**

embedding space by considering the contextualized embedding matrix as the negative pair, unlike Loss1 which only considers embeddings in the same batch as negatives (in-batch negatives). However, Loss2 show lower performance than Loss1 as in the case where the model considers a single token embedding as a positive pair, the model considers the rest of the token embeddings in the same title as the negative pair. As the token embeddings in the same title are matched with the same query, such a training method seems to make the model confused and leads to bad performance. Thus when considering a single token embedding as positive pair (Loss1 or Loss2), it is better to consider only the embeddings in the same batch as negatives (in-batch negatives) rather than on all the token embeddings (Contextualized Embedding Matrix) as there is a low possibility of the model to have two different token embeddings of the same title in a batch.

(3) Contextualized Embeddings Size As saving all contextualized token embeddings to use as the vocab embedding matrix requires a large storage footprint ($\approx$ 148GB), we reduce the number of token embeddings by clustering and saving only the k centroid embeddings for each token (Section 3.5). Figure 3 shows the effect of the maximum number of clusters for each token (k) on the performance. Models with a $k = 5$ (maximum of five different contextualized token embeddings for each token) show the highest performance and having k smaller or larger than five decreases the performance. We hypothesize that the performance of models with $k < 5$ degrades because the number of the embeddings is too small to contain all different contextual meanings of the token and thus will be closer to vanilla token embedding. In contrast, the performance of models with $k > 5$ decreases because the search space of each generation step is too large and the parametric space of the model becomes too fine-grained which might distract the model.

(4) Longer Context To see how informative the document context (length of the context) affects performance, we compared the performance of BASE, BASE-Short, and GENRE*. GENRE*, which uses vanilla vocab embedding as the target embedding, has to depend solely on the information encoded in its own parameters (the parametric space of the generative retrieval model). On the other hand, BASE and BASE-Short can depend on not only the parametric space of the generative retrieval model but also the non-parametric space of corpus information embedded in the contextualized target embedding. By utilizing the contextualized target embedding, the model can know in which context the token is used and discern documents with different contexts.

Although both BASE and BASE-Short utilize contextualized target embeddings, the contextualized target embedding of BASE-Short contains shorter context information compared to that of BASE. Therefore, BASE-Short fails in cases where the document content is necessary to retrieve the target sequence successfully. It is difficult for the model to predict the target without the help of the document content about what information is in the document or what relationship exists between the query and the target sequence. We can see from the table (Table 4) that BASE successfully retrieves as such information is embedded in the contextualized target embeddings whereas BASE-Short fails as it does not contain the document content in its embeddings. Also, Table 10 shows that there is a correlation between the performance and how much contextual information is encoded in the non-parametric space.

Characteristics by different CE Encoder We compare the contextualized token embeddings of BASE, ASYNC, and CONTRA[16] For 1000 cluster embeddings, we check the rate of the same token among the top-5 embeddings similar to the corresponding embedding. BASE shows the lowest rate of 50%. ASYNC and CONTRA show a similarly high rate of 70%. The rate tends to increase as N increases in ASYNC. Such results suggest that as

[16]We analyze the CE Encoder of step2 in CONTRA and last replace CE Encoder for ASYNC.

the same token has a similar lexical meaning, it is better to have a relatively similar meaning. However, as the performance increases as a single token are matched to multiple token embeddings till $k = 5$, it is also important to have slightly different meanings depending on the surrounding context. When checking which corpus bundles are bound to the same cluster, all three tend to depend on which *position* of text the token is placed on and the *meaning* of surrounding tokens. For example, when we cluster a token “Bee" into five clusters, it tends to group in: (1) word related to a person’s name where “Bee" appears in the middle of the name (Edmund Beecher Wilson), (2) word related to food where “Bee" appears at the front of the word (Beef Jerkey), (3) “Bee" with Spelling Bee (The 25th Annual Putnam County Spelling Bee) or the insect bee (Honey to the Bee) that appears at the end of a word, (4) word related to music where “Bee" appears at the front of the word (Honey to the Bee), (5) word related to film or TV series where “Bee" appears near the end (Queen Bees (TV series)). Such tendencies are shown in all three models.

Clustering over total embeddings To understand the spatial properties of the contextualized embeddings, we conducted a qualitative analysis on the embeddings, by performing k-means clustering over the total contextualized token embeddings of BASE (CE Encoder is the encoder of T5-large). Specifically, we clustered 36 million token embeddings, obtained from CE Encoder, into 117,508 clusters[17] using the FAISS k-means library (Johnson et al., 2021).

First, we randomly sampled 100 tokens, and for each token, we calculated the portion of the contextualized embeddings that belong to the top 10% of the clusters which contain the most embeddings of the token. As a result, on average 67.6% of the embeddings of a token are contained in the 10% of the clusters which contain the token, with a standard deviation of 22.7. This indicates that most of the tokens are concentrated in a few spatial regions, while the others are spread over many different areas.

To get a deeper insight into the spatial properties of the embeddings, we picked two tokens, “Lincoln" and “Squad" and visualized some of the clusters that contain the tokens(Table 12). For each cluster, the tokens belonging to the cluster and their corresponding document names are shown. In (Table 12), at most 20 documents are shown for each token and only 4 tokens are shown in cluster 3 and 4 for simplicity. The first and second examples show the case that a cluster is composed of only a single token, as mentioned above. Interestingly, all of the corresponding documents of the first cluster are related to Lincolnshire, a county of England. Similarly, the tokens in the second cluster are related to the documents about sports (usually football) squads. On the other hand, the third and fourth examples show the other case that a cluster contains only a few tokens that we are interested in. The members of the third cluster are related to the middle names, and a few embeddings of the token “Lincoln" is contained in this cluster since there are some Wikipedia documents of the people whose middle name is Lincoln. Likewise, the fourth cluster consists of the embeddings which are related to the name of music albums(usually hip-hop and rock), where some of them are produced by the group named “Blazin’ squad", for example. These examples show how expressive can the contextualized embeddings be compared to the vanilla token embeddings; in this case, it is hard to expect that this various context-dependent information of a token can be sufficiently encoded into a single token embedding.

In summary, the results show that the contextualized embeddings corresponding to the same token are mapped to many different regions of the embedding space, depending on its context. This implies that the contextualized embeddings successfully acquired the contextual information of the corresponding documents, highlighting the effectiveness of utilizing contextualized embeddings for generative retrieval.

[17]The number of the clusters is same as the number of the tokens in contextualized embedding matrix, hence same as the number of the clusters we used in 3.5.

Table 12: Examples of clusters when clustering over total contextualized token embeddings when CE Encoder is the encoder of T5-large.

Cluster	Token	Documents
1	**_Lincoln**	Moulton, Lincolnshire / Belton, North Lincolnshire / Walcott, Lincolnshire / Wrangle, Lincolnshire / Swineshead, Lincolnshire / Leverton, Lincolnshire / Kirton, Lincolnshire / Benington, Lincolnshire / Bicker, Lincolnshire / Dyke, Lincolnshire / Hilldyke, Lincolnshire / Waltham, Lincolnshire / Reepham, Lincolnshire / Bradley, Lincolnshire / Allington, Lincolnshire / Donington, Lincolnshire ettleton, Lincolnshire / Panton, Lincolnshire / Beckingham, Lincolnshire / Bigby, Lincolnshire / ...
2	**_Squad**	Field hockey at the 2000 Summer Olympics – Men's team squads / Field hockey at the 2004 Summer Olympics – Men's team squads / Field hockey at the 1996 Summer Olympics – Men's team squads / Football at the 2000 Summer Olympics – Men's team squads / Football at the 1996 Summer Olympics – Men's team squads / List of Queensland rugby league team squads / Football at the 2006 Lusophony Games – Men's team squads / List of current AFL team squads / Football at the 1912 Summer Olympics – Men's team squads / List of New South Wales rugby league team squads / Football at the 1996 Summer Olympics – Women's team squads / Football at the 1988 Summer Olympics – Men's team squads / Football at the 1984 Summer Olympics – Men's team squads / Football at the 1976 Summer Olympics – Men's team squads / Football at the 1900 Summer Olympics – Men's team squads / Football at the 1904 Summer Olympics – Men's team squads / Football at the 1908 Summer Olympics – Men's team squads / Football at the 1992 Summer Olympics – Men's team squads / Football at the 1980 Summer Olympics – Men's team squads / Football at the 1972 Summer Olympics – Men's team squads / ...
3	**_Lincoln**	William Lincoln Garver / Albert Lincoln Washburn / Charles Lincoln Edwards / Thomas Lincoln Casey Sr. / James Lincoln Collier / Abraham Lincoln Lewis / Earl Lincoln Poole / George Lincoln Goodale / George Lincoln Burr / Abraham Lincoln Keister / Elmer Lincoln Irey / Walter Lincoln Hawkins / Abram Lincoln Harris / Abraham Lincoln DeMond / Thomas Lincoln Tally / Abraham Lincoln Filene / Mary Lincoln Beckwith / Frederick Lincoln Emory / Howard Lincoln Hodgkins / Oliver Lincoln Lundquist / ...
	_Levi	John Levi Marti / John Levi Sheppard / Moses Levi Ehrenreich / Nathaniel Levi Gaines / Harry Levi Hollingworth / Thomas Levi Whittle / Austin Levi Fraser / George Levi Crane / Olin Levi Warner
	_Luke	Milledge Luke Bonham / Henry Luke Orombi / Henry Luke White / Vincent Luke Palmisano / George Luke Smith / Henry Luke Bolley / Mary Luke Tobin / James Luke Prendergast / John Luke Lowther / Jerry Luke LeBlanc / Thomas Luke Msusa / Robert Luke Deakin / Joseph Luke Cecchini
	_Lane	Carroll Lane Fenton
	...	
4	**_Squad**	True Story (Terror Squad album) / The Album (Terror Squad album)
	_Angel	Covenant (Morbid Angel album) / Domination (Morbid Angel album) / The Art of Dying (Death Angel album) / Act III (Death Angel album) / Heretic (Morbid Angel album)
	_Butterfly	Heavy (Iron Butterfly album) / Metamorphosis (Iron Butterfly album) / Ball (Iron Butterfly album)
	_Flip	Flip-flop (electronics) / Flipper (anatomy) / Respect Me (Lil' Flip album) / The Leprechaun (Lil' Flip album)
	...	

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☒ A2. Did you discuss any potential risks of your work?
all outputs are from a given corpus set, in which the possible risk can be controlled by human

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1 (Introduction)

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Section 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B.1

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix B and Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Appendix B

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B.2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.